Managing More With Less Doesn’t Have To Be Impossible

We will always be asked to do more with less. Finance is asked to produce more and better analytics with less people. Sales people are asked to produce more in a weakening economy with less marketing dollars, and yes, groups that manage Essbase environments are asked to produce and manage more data/applications with shrinking resources.

Back in the Day

In a prior life, I used to manage a group responsible for managing the Essbase environment used to produce all the reporting for the group. It generated about 70% of the revenue for Bank One (now Chase). We delivered all the reporting, budgeting, and forecasting applications. It included nearly 2 TB of data (pre ASO) on four servers that included more than 50 databases. All the typical technologies were employed. A large number of filters existed to maintain security. Many of the applications were linked together with several types of partitions. Data was loaded daily, weekly, and monthly. SQL Server was used for all the ETL processes, and we completed the development and performed all the maintenance with four people.

The only way the group could be effective in developing and enhancing applications, was to eliminate our effort spent on typical production activities. With the number of applications and the frequency they were updated (daily, weekly, or monthly), communicating this information to the more than 250 users was also a large time commitment.

The Solution

We built custom applications using the Essbase API to not only automate the tasks, but also notify the appropriate person if there was an error. This included everything from data loads, application builds, ETL processing, nightly data exports, repetitive calculations, and every other aspect of the maintenance. We even automated the validation of the data during the load process. Data loaded to the ETL layer was compared to the ETL Export. After the data was loaded to the Essbase application, we automated Excel data retrieves and compared them back to the ETL data exports. We effectively eliminated any effort it took to maintain the environment unless an error occurred that required attention. This was the ONLY way we could keep our heads above water.

We chose the API because it is so robust. It has most of the Maxl functions. It introduces the ability to check for errors at any point in the process, and can take the appropriate steps to resolve. No manual intervention was required. The same application can interact with the ETL layer, send text messages or pages, email administrators and users, and update web pages with statuses that the users can see (like the state of the load process, calculation status, etc.).

This solution may be overkill for very small implementations of one or two applications. But, don’t underestimate its importance in medium to large-scale operations. It minimizes costs, reduces errors, provides a better user experience, and minimizes delays in new development.

I highly recommend investigating how this would work for your group. Although I used the Visual Basic API, there are also libraries for C and Java.